FACULTY OF COMPUTING
SEMESTER II, SESSION 2024/2025

BACHELOR OF COMPUTER SCIENCE (BIOINFORMATICS)

SECB3203 PROGRAMMING FOR BIOINFORMATICS - SECTION 01

PROJECT
ALZHEIMER'S DISEASE PREDICTION

GROUP MEMBERS	MATRIC NO
TAN ZHAO HONG	A23CS0188
CHIN PEI WEN	A23CS0065
KOO XUAN	A23CS0300

LECTURER'S NAME : DR. SEAH CHOON SEN

SUBMISSION DATE : 03 DECEMBER 2025

Table of Contents

3.0 Flowchart of the Proposed Approach

3.1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an essential step in understanding the structure, characteristics, and patterns within the dataset before building machine learning models. In this project, EDA is conducted using Pandas, NumPy, Matplotlib, and Seaborn to explore the Alzheimer's Disease dataset. The analysis focuses on descriptive statistics, data grouping, analysis of variance (ANOVA), and correlation analysis.

3.1.1 Descriptive Statistics

Descriptive statistics are used to summarize the main characteristics of the dataset numerically. Using the df.describe() function from Pandas, statistical measures such as mean, standard deviation, minimum, maximum, and quartiles are calculated for all numerical attributes.

```python
    # Statistical summary
    print("\nStatistical Summary:")
    print(df.describe())

    return df
```

```
[2] Performing Exploratory Data Analysis...

Statistical Summary:
             Age  EducationLevel          BMI      Smoking  PhysicalActivity  ...  Disorientation  PersonalityChanges  DifficultyCompletingTasks  Forgetfulness    Diagnosis
count  2149.000000     2149.000000  2149.000000  2149.000000       2149.000000  ...     2149.000000         2149.000000                2149.000000    2149.000000  2149.000000
mean     74.908795        1.286645    27.655697     0.288506          4.920202  ...        0.158213            0.150768                   0.158678       0.301536     0.353653
std       8.990221        0.904527     7.217438     0.453173          2.857191  ...        0.365026            0.357906                   0.365461       0.459032     0.478214
min      60.000000        0.000000    15.008851     0.000000          0.003616  ...        0.000000            0.000000                   0.000000       0.000000     0.000000
25%      67.000000        1.000000    21.611408     0.000000          2.570626  ...        0.000000            0.000000                   0.000000       0.000000     0.000000
50%      75.000000        1.000000    27.823924     0.000000          4.766424  ...        0.000000            0.000000                   0.000000       0.000000     0.000000
75%      83.000000        2.000000    33.869778     1.000000          7.427899  ...        0.000000            0.000000                   0.000000       1.000000     1.000000
max      90.000000        3.000000    39.992767     1.000000          9.987429  ...        1.000000            1.000000                   1.000000       1.000000     1.000000

[8 rows x 29 columns]
```

3.1.2 Basic of Grouping

Grouping analysis is performed using the groupby() function to examine how different features behave across groups.

Age Grouping

The Age attribute is grouped into three categories:

Young (≤ 60 years)

Middle-aged (61–75 years)

Old (> 75 years)

This binning approach helps simplify analysis and reveals trends related to age categories. The frequency of each age group is visualized using bar charts, allowing comparison of population distribution across age ranges.

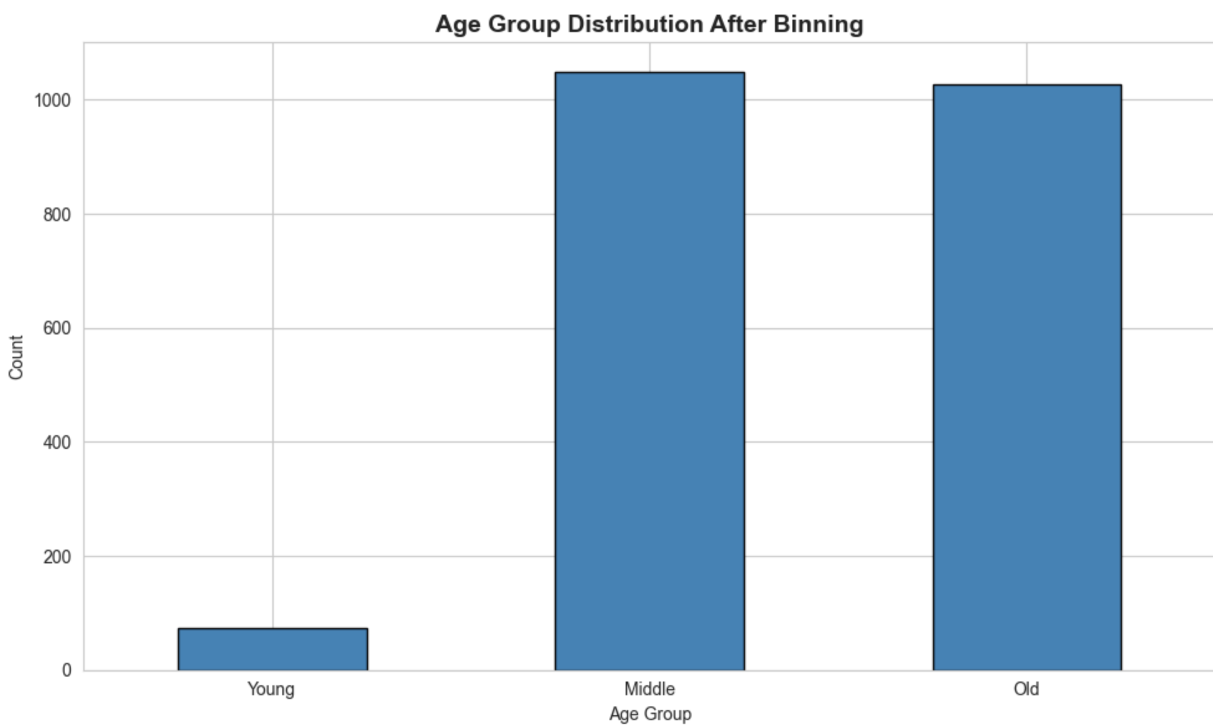
Grouping helps highlight how different demographic segments contribute to Alzheimer's diagnosis patterns.

```python
    if 'Age' in X.columns:
        print("\n1. Age Binning:")
        X['Age_binned'] = pd.cut(
            X['Age'],
            bins=[0, 60, 75, 100],
            labels=['Young', 'Middle', 'Old']
        )

        # Show binning results
        print("\n   Sample of Age Binning:")
        print(X[['Age', 'Age_binned']].head(10))

        print("\n   Age Group Distribution:")
        print(X['Age_binned'].value_counts())

        # Visualize
        plt.figure(figsize=(10, 6))
        X['Age_binned'].value_counts().sort_index().plot(kind='bar', color='steelblue', edgecolor='black')
        plt.title('Age Group Distribution After Binning', fontsize=14, fontweight='bold')
        plt.xlabel('Age Group')
        plt.ylabel('Count')
        plt.xticks(rotation=0)
        plt.tight_layout()
        plt.savefig('age_binning.png', dpi=300)
        plt.show()
```



3.1.3 ANOVA

Analysis of Variance (ANOVA) is applied to determine whether there are statistically significant differences between numerical features across different diagnosis groups.
ANOVA evaluates:
Whether the mean values of numerical features differ significantly between Alzheimer's and non-Alzheimer's patients.
If observed differences are due to actual group effects rather than random variation.
This step supports feature relevance assessment, helping identify which attributes are more influential in distinguishing diagnosis outcomes.

```
from scipy.stats import f_oneway

print("\n--- ANOVA Test (Age vs Diagnosis) ---")

group_0 = df[df['Diagnosis'] == 0]['Age']
group_1 = df[df['Diagnosis'] == 1]['Age']

f_stat, p_value = f_oneway(group_0, group_1)

print(f"F-statistic: {f_stat}")
print(f"P-value: {p_value}")
```

```
--- ANOVA Test (Age vs Diagnosis) ---
F-statistic: 0.06467450176025781
P-value: 0.7992790224412292
```

3.1.4 Correlation

Correlation analysis is conducted using the Pearson correlation coefficient through the corr() function in Pandas.

Correlation Matrix
A correlation matrix is generated for all numerical features to measure the strength and direction of linear relationships between variables.

Heatmap Visualization
A heatmap is plotted using Seaborn to visualize correlation values:

- Positive correlations indicate that variables increase together.

- Negative correlations indicate an inverse relationship.
- Values close to 1 or -1 represent strong correlations, while values near 0 indicate weak relationships.

This analysis not only helps to identify highly correlated features that may cause multicollinearity. It also helps understand relationships between predictors and the target variable and support feature selection decisions for machine learning models.

```python
# Correlation heatmap for numerical features
numerical_cols = df.select_dtypes(include=[np.number]).columns
if len(numerical_cols) > 1:
    plt.figure(figsize=(14, 10))
    correlation_matrix = df[numerical_cols].corr()
    sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm',
        center=0, square=True, linewidths=1)
    plt.title('Correlation Heatmap of Numerical Features')
    plt.tight_layout()
    plt.show()
```